FIGS. 18A and 18B illustrate the preferred process for executing filters when a dynamic document pointer is created or revised. DDs have three sets of filters: filters for determining what data to display on the screen, filters for determining what data to print on a report, and filters to determine which changes to send to targeted production data sources. All three types of DD filters work almost identically to MRT filters. The calculation logic is the same except DDPs correspond to MRPIs, Panes correspond to MRPFs. MRT filters are always executed prior to DD filters. Unlike MRT filters, DD filters are not all executed at the same time for a single DDP. When first constructing a DD's DDPs to minimize the number of MRPFs processed in arriving at the pointers for a pane, the present invention categorizes filters into four categories and executes them at different times during the creation and updating of DDPs: 1) Prejoin filters are executed on a pane's MRPF's MRPIs prior to joining them with the other MRPIs for the pane. Filters that have operands on only the fields in the PRTs for the MRPF are in this category. Also filters which meet the first criteria and also use a calculated field from the MRPF and lead PRT for the MRPF is in the current pane or any parent pane are in this category; 2) Post join filters are executed after all of a pane's MRPIs are joined together with its parent pane's MRPIs. Filters that meet the above criteria but used fields from more than one MRPF are in this criteria; 3) Post summarize filters are executed after the joined MRPIs are summarized into Pane Pointers. Filters which do not use calculated fields from DD functions are not in either of the above two categories; and 4) Post function filters are executed after the select functions but prior to the display filters. All filters that use calculated fields that are the result of DD functions are in this category.

The process begins by reading any new or changed DDP in step 380. Then a filter from the DD definition is read in step 381. Next, in step 382, an operand is tested, and the result is saved. Since all filter operations are on fields in the DD, each DD can be fully tested to see if it should be filtered out by reading only the DDP's calculated fields. For improved processing efficiency, the present invention executes all of a DDPs filters one after the other. The operation tests are simple Boolean tests to see if the field's value equals the value stored in the list (Dynamic Document) specified. The present invention stores the result (true or false) of each operation. The method then determines if this operand is the last operand to test in step 383. If not, the process returns to step 382 to test the next operand. If all the operands have been tested, then the process continues to step

384. In the step 384, the method looks at the results of each operand to determine if the full filter results in a pass or fail. If the DDP passes the filter, the method proceeds to step 386 where the method appends to the DDP a flag for each filter indicating the filter includes that particular DDP. If the DDP fails the filter, the method jumps to step 388 to evaluate the filter type. In step 388, further tests are performed to determine if the filter is a delete type. If the filter is a delete type, the DDP and its children DDPs are deleted in step 389. Any DDP which has any parent DDP which are excluded is itself excluded. If the filter is not a delete type, then the flag is set to exclude for this filter in the current DDP's filter affects list in step 390. Any DDP that is excluded by any active filter is not displayed in the currently open DDs. Next, step 391 determines whether there are additional filters to be processed. If so the method loops to 381, otherwise the method tests whether this is the last MRPI to be processed in step 392. If there are more MRPIs to process, the method returns to step 380, otherwise, the process is complete.

Finally, it should be understood that the present invention also uses conventional sorting and display techniques. Sorting occurs dynamically when the DDPs are joined to the PRTs and displayed as pane rows. Sorting is done pane by pane in the same manner employed by conventional report write and query tools. The rows in a pane can be sorted by any of PRT and/or calculated fields in the pane. The user specifies the order in which each field sort is executed and whether to use an ascending or descending order. The user can also specify which, if any, sort levels should have total lines and what group by operand to use for each field for each level. The user can also have the present invention suppress displaying the row details and just display the totals. When the user selects (highlights) a total row in a pane, that pane's child panes display the rows for all the parent pane's detail rows that comprise the selected total row.

The display and reporting component of the present invention works similarly to many conventional query and reporting tools. By the time the present invention hands over the data to be displayed, it has already organized it, executed its calculations, filtered and sorted it. The present invention hands over one record (a DDP and its corresponding PRIs) to the screen forms display tool. All that remains is to place the record's values in the proper fields. Display rules entered by the user control the appearance of the panes, positioning the format of the PRT's fields, and what type of maintenance users can do to actual field values.

Users can open as many DDs at once as their hardware supports. Each DD can be autonomous or any DD can be linked to another. To link DD all the user need do is specify which on PRTs within the two DDs the link occurs. The two PRTs must be hooked together, or the PRT in both DD must be identical. At least one of the DDs must be linked at its top pane and a core linked DD must be designated as the controlled DD. Any DD can have only one controlling DD and circular controls are not permitted. When the user selects a new record instance in the controlling DD, the controlled DD is updated to display the record instance(s) that are hooked to the new controlling record instance. The present invention does this by executing the "find" function on the controlled DD; the find parameter is the foreign key or pointer of the controlling DD's current record type instance.

The present invention can save all the contents of memory it is currently using as a single file. If the user wishes to leave the terminal for a while, the user can save it and then recall it whenever required. This save is not shared with other users. Multiple copies of a session can be made and later modified just like as can be done with spreadsheets an word processors. In a multiple user environment, the changes to PRTs are stored in a relational database or XML documents and multiple users can access those changes. The definitions of MRTs, hooks, DDs, functions, filters, and sorts are also stored in the same database in special tables or XML documents setup by the present invention. Optionally MRPIs and DDPs can also be stored and maintained by the present invention in the same relational database or XML documents. It is useful to store these pointers in the repository because they do not need to then be regenerated for each user when he/she resumes working on the shared model.

While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided. For example, there may be other embodiments for the method of executing the functions, filters and sorts used in FIG. 4 in addition to those described above. Similarly, there may be other embodiments for the particular structure used for the MRTs, the pointer families, and DDs. These